Programação Orientada a Objetos

JAVA

WRITE ONCE, COMPILE ONCE AND RUN ANYWHERE!

Programação Orientada a Objetos

História

- Em 1991 a Sun Microsystems, de olho no crescente mercado de dispositivos eletrônicos inteligentes, voltados ao consumidor financiou um projeto interno (Green Project);
- A ideia era produzir uma linguagem de computador reduzida e simples e que gerasse um código eficiente para ser utilizada em dispositivos com algumas restrições:
  - Pouca memória;
  - Diferentes CPUs;
    - O software produzido não poderia se limitar a uma única arquitetura.
- James Gosling (chefe da equipe) batizou a linguagem de Oak.

Programação Orientada a Objetos

História (cont..)

- O nome Oak já era patenteado então a linguagem foi batizada de Java em homenagem a Ilha de Java que produzia o café que a equipe da Sun consumia.
- O mercado de dispositivos eletrônicos não teve um crescimento tão interessante conforme previsto e o Green Project perdeu força;
- Na mesma época há uma explosão de popularidade da *World Wide Web* e este parece ser um mercado potencial para a linguagem Java;
- Em 1995 Java é anunciado formalmente.
- Em 2009 a Oracle adquire a SUN por US$ 7,4 bilhões e é quem mantém o Java atualmente.

Programação Orientada a Objetos

Características

- Linguagem simples e orientada a objetos;
- Subconjunto de C++
  - Reduzido número de palavras reservadas e grande poder de expressão;
  - Construções complexas e desnecessárias que não fazem parte do núcleo mínimo exigido para uma linguagem orientada a objetos são inicialmente eliminadas ...
    - Instruções pré-processadas;
    - Herança múltipla;
    - Sobrecarga de operadores;
    - Programação genérica

Programação Orientada a Objetos

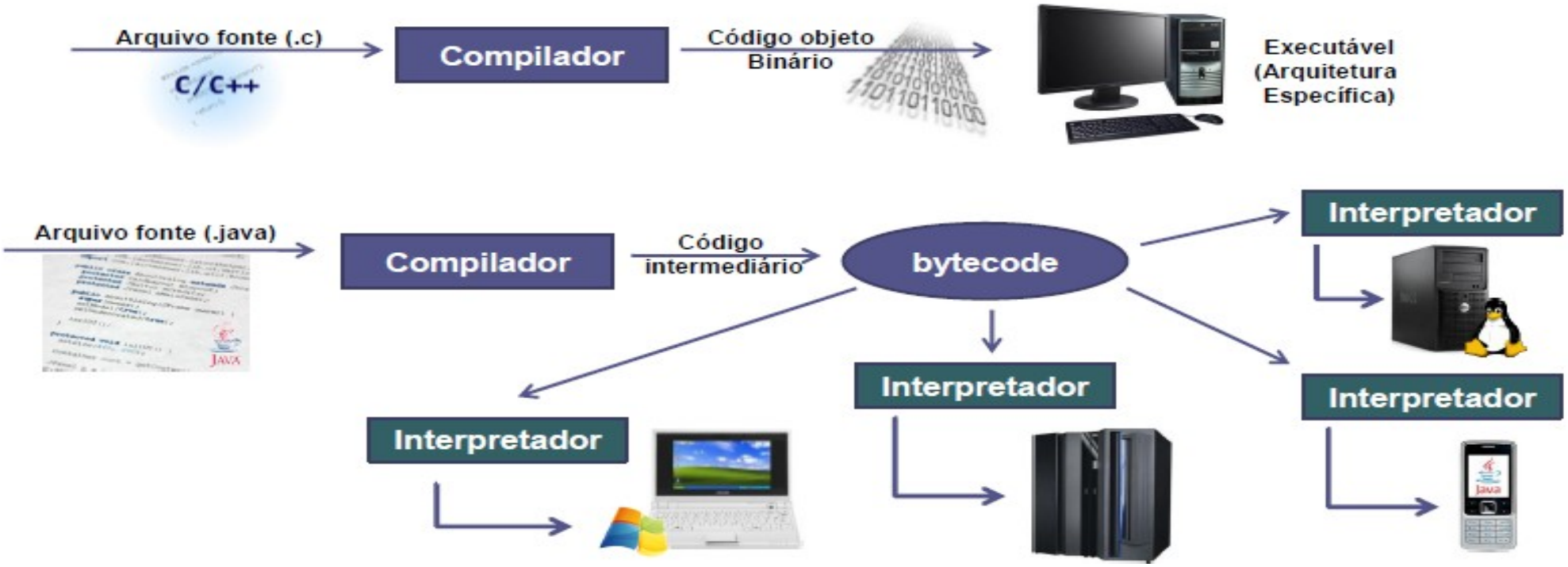
Características (cont..)

- Fortemente tipada
  - Ampla verificação e erros e checagem de tipos em tempo de compilação;
- Inexistência do conceito de ponteiros
  - O que existem são referências a objetos;
- Gerenciamento automático de memória
  - *Garbage Collection* – Coletor de Lixo

Programação Orientada a Objetos

Características (cont..)

- Independente, Interpretada e Portável



Programação Orientada a Objetos

Etapas do desenvolvimento

Programação Orientada a Objetos

Vantagens

- Independência de Hardware e de Sistema Operacional;
- Isola as aplicações do Sistema Operacional;
- Gerenciamento de recursos;
- Separação de servidores;

Slogan : **WRITE ONCE, RUN ANYWHERE**

Programação Orientada a Objetos

Versões:

- Java 1.0 e 1.1,
- Java2 (Java 1.2)
- Java2 1.3, Java2 1.4
- Java 5 (Java2 1.5)
- Java 6
- Java 7
- Java 8

Baixar - http://www.oracle.com/technetwork/java/

Programação Orientada a Objetos

Edições:

- ***JSE – Java Standard Edition:*** para aplicações *desktop*;
- ***JEE – Java Enterprise Edition:*** para o desenvolvimento de aplicações corporativas;
- ***JME – Java Micro Edition:*** para desenvolvimento de aplicações com recursos limitados;

Programação Orientada a Objetos

Antes de começar..

- `.java` = código fonte
- `.class` = código intermediário (bytecodes)
- JRE = Java Runtime Environment
  - Máquina Virtural Java (JVM = Java Virtual Machine)
- JDK = Java Development Kit (Kit de Desenvolvimento)
  - `javac` – compilador
  - `java` – interpretador de bytecodes
  - `javadoc` – gerador de documentação
  - `jdb` – debugger
  - `jar` – ferramenta para gerência de *Java Archives*
  - `javap` – disassembler (fornece um protótipo das classes encontradas no arquivo .class);

Programação Orientada a Objetos

JDK X JRE

Programação Orientada a Objetos

Instalação:

- baixar o JDK da edição JSE
- Instalar, configurar
  - **JAVA_HOME** : local de instalação do Java no SO
    - *JAVA_HOME=C:\Program Files\Java\jdk1.7.0_45*
  - **PATH** : caminho que o interpretador procura os executáveis
    - *PATH=%PATH%;C:\Program Files\Java\jdk1.7.0_45\bin*
  - **CLASSPATH** : caminho onde a JVM procura as classes e bibliotecas
    - *CLASSPATH=C:\Program Files\Java\jdk1.7.0_45\lib;,;*

- Teste: execute **“java -version”** e **“javac -version”** no prompt de comando

Programação Orientada a Objetos

Conceitos Básicos e Sintaxe da Linguagem Java

- Todo programa Java fica em uma Classe.
- Os programas iniciam com o método main().
- Blocos de comandos limitados por chaves “{“ e “}”.
- Instruções terminadas por “;”.
- Java é *case sensitive.*

Programação Orientada a Objetos

Primeiro Programa Java

```
class MeuPrograma {
    public static void main(String[] args) {
        System.out.println("Minha primeira aplicação Java!");
    }
}
```

MeuPrograma.java

Programação Orientada a Objetos

Método ***main(String[ ] args)***

- Permite que uma classe seja executada (normalmente pelo menos uma das classes da aplicação precisa ser executada);

- Quando uma aplicação java é iniciada, a JVM localiza e chama o método main() que recebe por parâmetro um vetor de objetos String representando os parâmetros de linha de comando;

  public static void main(String[ ] args)

  - args => passagem de argumentos na linha de comando para o programa Java.
  - A execução da aplicação Java continua até que o método main() termine de executar todos as suas instruções.

Programação Orientada a Objetos

Comentários:

- // - comenta a linha
- /* */ - comenta um bloco

```
{
  // este é um bloco de programação
   int a=10;
}
```

```
{
   int a=10;
   int b;
   b=a*2;
   /* a partir deste ponto, deve-se começar a
exibir
   os resultados na tela do usuário */
}
```

```
{
   int a=10;
   int b;
   int c;
   int soma;
   int x;
   b=a*2;
   c = a*2 + b*2;
/* soma = a + b + c;
   a += b;
   ++b;
   ++c;  */
   x = ((a + b + c) * 2 ) / 4;
}
```

Programação Orientada a Objetos

Tipos Primitivos

Tipo	Descrição	Faixa de Valores
boolean	Valor booleano	True ou false
char	Único caracter representado em 16 bits	0 a 65535
byte	Inteiro de 8 bits, com sinal	-128 a 127
short	Inteiro de 16 bits, com sinal	-32768 a 32767
int	Inteiro de 32 bits, com sinal	-2147483648 a 2147483647
long	Inteiro de 64 bits, com sinal	-9223372036854775808 a 9223372036854775807
float	Ponto flutuante de precisão simples e 32 bits, com sinal	1.4012984632481707e-45 a 3.4028234663852860e+38
double	Ponto flutuante de precisão dupla e 64 bits, com sinal	4.94065645841246544e-324 a 1.79769313486231570e+308

Programação Orientada a Objetos

Declarando Variáveis

```
public class Exemplo{

        public static void main(String[] args){
            byte a = -128;
            byte b = 127;
            short c = -32768;
            short d = 32767;
            int e = -2147483648;
            int f = 2147483647;
            long g = -9223372036854775808L;
            long h = 9223372036854775807L;
            float i = -100.4345f;
            float j = 123243.4345f;
            double k = -3123.434354;
            double l = 321321.3123435;
            boolean m = false;
            boolean n = true;
            char o = 'a';
            char p = '4';
            char q = '?';
    }
}
```

Programação Orientada a Objetos

Caracteres Especiais

- \' - apóstrofo
- \” - aspas
- \\ - barra invertida
- \b - backspace
- \n - nova linha
- \t - tabulação

Programação Orientada a Objetos

Casting e Promoção

- Atribuição de valores incompatíveis

```
int i = 5
double d = i   // compila

double d = 5.5
int i = d   // não compila
int i = (int) d  // compila
```

Programação Orientada a Objetos

Casting e Promoção

PARA: / DE:	byte	short	char	int	long	float	double
byte	----	*Impl.*	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
short	(byte)	----	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
char	(byte)	(short)	----	*Impl.*	*Impl.*	*Impl.*	*Impl.*
int	(byte)	(short)	(char)	----	*Impl.*	*Impl.*	*Impl.*
long	(byte)	(short)	(char)	(int)	----	*Impl.*	*Impl.*
float	(byte)	(short)	(char)	(int)	(long)	----	*Impl.*
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

Programação Orientada a Objetos

Operadores:

- Aritméticos (+, -, *, /, %)
- Relacionais (>, >=, <, <=, ==, !=)
- Lógicos (&&, ||, !)

- Atribuições (=,+=,-=,*=,/=,%=)
- Incrementos (++variável, variável++)
- Decrementos (--variável, variável--)

Programação Orientada a Objetos

Constantes em Java - **final**

```
{
...
final int a = 2;
a = 3;
}
```

não compila

Programação Orientada a Objetos

Outro Programa Java

```
class Soma {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int soma = a + b;
        System.out.println ("a + b = " + soma);
    }
}
```

Soma.java

Programação Orientada a Objetos

Estruturas de Controle

Condicionais:

- **if**

```
if (expressaoLogica) {
    comandos;
}
```

- **if / else**

```
if (expressaoLogica) {
    comandos;
}
else {
    outros_comandos;
}
```

Programação Orientada a Objetos

Condicionais (cont..):

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && amigoDoDono == false) {
    System.out.println("Não pode entrar");
}
else {
    System.out.println("Pode entrar");
}
```

Programação Orientada a Objetos

Condicionais (cont..)

- **swich**

```
swich(variavel) {
    case valor1 : comandos1;
                  break;
    case valor2 : comandos2;
                  break;
    case valor3 : comandos3;
                  break;
    ....
    default : comandosdefault;
}
```

Programação Orientada a Objetos

Condicionais (cont..)

- **swich**

```
swich(tipo) {
    case 1 : System.out.println("Solteiro");
             break;
    case 2 : System.out.println("Casado");
             break;
    case 3 : System.out.println("Separado");
             break;
    case 4 : System.out.println("Disquitado");
             break;
    case 5 : System.out.println("Viúvo");
             break;
    ....
    default : System.out.println("Indefinido"); ;
}
```

Programação Orientada a Objetos

Repetição:

- **while**

```
while (expressaoLogica) {
    comandos;
}
```

- **do / while**

```
do {
    comandos;
} while (expressaoLogica)
```

- **for**

```
for (expressaoInicial; expressaoLogica; incremento) {
    comandos;
}
```

Programação Orientada a Objetos

Repetição (cont..):

```
int i = 0;
while (i < 10) {
        System.out.println(i);
        i = i + 1;
}

for (i = 0; i < 10; i = i + 1) {
        System.out.println(i);
}
```

Programação Orientada a Objetos

Controle de Repetição:

- **break**

  Força a saída de um comando de repetição ou do *switch*

- **continue**

  Força o início da próxima iteração de um comando de repetição

Programação Orientada a Objetos

Controle de Repetição:

```
for (int i = x; i < y; i++) {
    if (i % 19 == 0) {
        System.out.println("Achei um número divisível por 19 entre x e y");
        break;
    }
}

for (int i = 0; i < 100; i++) {
    if (i > 50 && i < 60) {   // ignoro os números entre 50 e 60
        continue;
    }
    System.out.println(i);
}
```

Programação Orientada a Objetos

Escopo de Variáveis:

- Nome dado ao trecho de código em que a variável existe e onde é possível acessá-la
- Depende de onde foram declaradas

Programação Orientada a Objetos

Convenções Java - ***Java Code Conventions***

- Padronização, facilidade de entendimento e manutenção

Exemplos:

- Nomes de classes;
- Declaração de variáveis;
- Nomes de variáveis e métodos;
- Comentários;
- Organização do código;
- Quebras de linha;
- Pacotes; etc..

http://www.oracle.com/technetwork/java/codeconv-138413.html

Programação Orientada a Objetos

- Identificadores

  Identificadores são os nomes que o programador utiliza para nomear suas classes, interfaces, métodos, variáveis, etc.

  - Nomes de classe com a primeira letra maiúscula,
    - exemplos:
      - `Teste`
      - `MinhaClasse`
  - Nomes de pacotes em letras minúsculas, exemplos:
    - `java.lang`
    - `java.pacoteTeste`

Programação Orientada a Objetos

- Identificadores (cont.)
  - Nomes de variáveis e métodos iniciando com letra minúscula, exemplos:
    - `variavel`
    - `minhaVariavel`
  - Nomes de constantes todas em letras maiúsculas, exemplos:
    - `PI`
    - `CONSTANTE_EXEMPLO`
  - Documentar cabeçalho de classes, atributos e métodos como comentários no estilo Javadoc /** ... */

Programação Orientada a Objetos

Exercício :

- Escrever, compilar e executar o primeiro programa

```
class MeuPrograma {
    public static void main(String[] args) {
    System.out.println("Minha primeira aplicação Java!");
    }
}
```

Programação Orientada a Objetos

Exercício (continuação):

- alterações no programa
  - Imprimir mais algumas linhas
  - Alteração no nome do arquivo
- Entendendo como funciona

Programação Orientada a Objetos

Declarando e usando variáveis:

Exercício:

```
class MostraIdadePesoSexo {
    public static void main(String[] args) {
        // imprime a idade, o peso e o sexo
        int idade = 20;
        double peso = 55.5;
        char sexo = 'F';

        System.out.println(“idade : ” + idade);
        System.out.println(“peso : ” + peso);
        System.out.println(“sexo : ” + sexo);
    }
}
```

Programação Orientada a Objetos

Exercícios de fixação:

- Criar um programa que dados 3 valores, mostra o maior e o menor;
- Criar um programa que liste números de 1 a 20;
- Criar um programa que liste o quadrado dos números de 1 a 20;
- Criar um programa que liste os números impares de 1 a 50;
- Criar um programa que liste a soma dos números de 1 a 50;
- Criar um programa que liste o fatorial dos números de 1 a 10;
- Criar um programa que calcule a fórmula e retorne X1 e X2;

$$x = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

Fórmula de Bhaskara

Programação Orientada a Objetos

Exercícios de fixação:

- Em uma empresa existem tabelas com os gastos mensais. Para fechar o primeiro trimestre, precisamos somar o gasto total. Sabendo que, em Janeiro, foram gastos 15000 reais, em Fevereiro, 23000 reais e em Março, 17000 reais, faça um programa que calcule e imprima o gasto total no trimestre. Siga esses passos:

  a) Crie uma classe chamada BalancoTrimestral com um bloco main, como nos exemplos anteriores;

  b) Dentro do main, declare uma variável inteira chamada gastosJaneiro e inicialize-a;

  c) Crie também as variáveis gastosFevereiro e gastosMarco, inicializando-as com os valores definidos;

  d) Crie uma variável chamada gastosTrimestre e inicialize-a com a soma das outras variáveis:

  e) Imprima a variável gastosTrimestre.

  f) Adicione código (sem alterar as linhas que já existem) para imprimir a média mensal de gasto, criando uma variável mediaMensal junto com uma mensagem. Para isso, concatene a String com o valor, usando "Valor da média mensal = "+ mediaMensal.